GIGAS Annotation Tool Guide

# Introduction

The GIGAS annotation tool uses image processing techniques to divide a given image into segments. Users can then associate any subset of those areas to one or more labels form a predefined taxonomy. The tool is designed to allow for efficient annotation of a large number of images using a combination of mouse clicks and keyboard presses. It also allows the user to annotated a single image or a sequence of images from a given directory. The annotations for each image are saved in a python dictionary pickle file that can be reloaded later by the tool to resume the annotation process.

Currently, the tool only uses the SLIC (Simple Linear Iterative Clustering) approach to to divide the image into *n* superpixels, where *n* is the desired number of segments and is selected by the user as explained below.

The tool is platform independent and uses Matplotlib library.

# Executing the GIGAS Annotation Tool

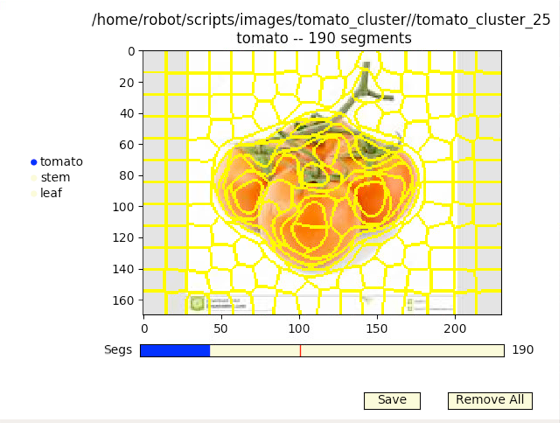
The tool is written in python 2.7. You will find it on the GIGAS group’s GitLab repository. The main program is annotate.py. To run it use the command:

python annotate.py –i *image\_file\_or\_dir* [-a *annotation\_file\_or\_dir]*

where:

* *image\_file\_or\_dir* is the full path to a dir containing the images to be annotated or a full path to a single image file.
* *annotation\_file\_or\_dir* is an optional argument that specifies the full path to the output directory where the annotations will be stored, or, if the input was a single image file, the full path to where the annotation file for that image is to be stored. If no output file/directory is given then the default output directory as defined in annotate.py will be used.

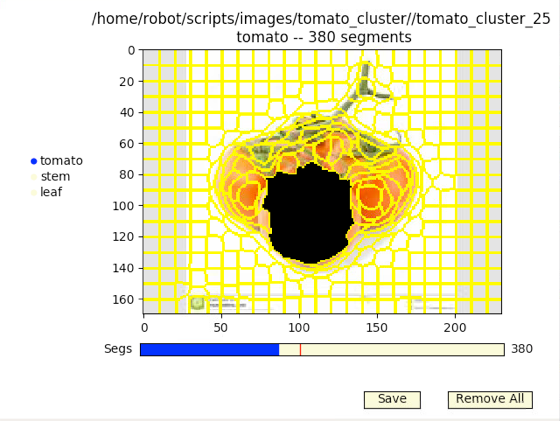
# Annotation Instructions



The application window consists of:

1. The image given by the user in the command line (or the first image from the given directory)
2. A clickable list of labels the user can use to annotate any segment in the image.
3. A slider that allows the user to vary the number of superpixels to generate for the image. Note that changing this number will cause all previously selected segments for that label to be reset.

## Steps to annotate an image

1. Start by selecting the label you will be annotating the segments with from the list of labels to the left of the image.
2. Move the **slider to select the desired number of segments**. IMPORTANT: if you change this value later for this label all the segments will be reset for this label.
3. Left click on a segments to **highlight the segment**, right click on it to undo the highlighting.
4. You can also **group segments together as one object**. For example, you can highlight several segments as shown below. To mark them as one tomato simply hit the space bar. This will mark the 5 segments as one object with the label tomato. The next segment you highlight will be considered part of the second tomato object.
5. The **Remove All** button resets all segments on the image for *the current label in the current image*.
6. If you are annotating a group of images, you can **navigate between images** through the left and right keyboard arrows. IMPORTANT: moving from one image to another will cause your current annotations to be saved before the new image is shown. This setting is meant to make annotating a large number of images easier.
7. You must remember to **save your annotations** before closing the application. There are several ways to save the annotations of the current image all of which save annotations for all labels for the current image:
   1. By clicking the Save button under the image.
   2. By pressing ESC on the keyboard. This will save the current image’s annotations and close the application.
   3. By navigating to the next or previous image.
8. To **exit the program without saving** your current annotations hit Ctrl+q.

# Annotation Details

Each image’s annotations are saved in a Python Pickle file. The annotation file’s name is the image file’s name appended with “\_annot.pkl”. The file contains a Python Pickle dump of the annotations dictionary for that image. The dictionary structure is as follows:

{ ***current image file name***:

{ ***label***:

**{‘annot\_mask’**: a SPARSE array containing the annotation information for *label*

**‘segParameters’**: A dictionary of necessary parameters for the segmentation method currently used. This is meant to accommodate further development of the tool where segmentation methods other than SLIC can be used. Currently this dictionary contains only *numSegments,* the number of segments used to create the superpixels.

**‘segType’:** The method used to segment the image. Currently we only have ‘KmeansSuperpixels’

**}**

}

}

Each file contains only one image file name key and all label keys where the user added annotations.

## Annotation Masks

The annotation masks are numpy arrays of the same size as the image. Each label has its own annotation masks. This allows us to label the same segment, or part of a segment, with multiple labels.

When a segment is clicked, the corresponding pixels in the mask are assigned a value equal to the object number being annotated at that moment. All unlabeled pixels have a value of 0.

For storage efficiency, the application saves masks as SPARSE arrays. This compresses the masks and reduces file size significantly (I have seen it reduced by 75%) The arrays are decompressed when they are loaded again.

# Future Work

There are several extensions we can add to enrich the tool, some more significant than others.

## Functionality Issues

1. **Reading the list of labels from a user supplied file**: we currently have a list of labels that is defined as a list in annotate.py We should have this list in a separate label list file. The file is loaded when the application is launched and the label list is created based on that file.
2. **Adding a new label from within the application:** an “add label” item should be added to the end of the radio button label list. When clicked, a new radio element is created and the user can type in the new label.
3. Adding a Reload Button: this allows the user to reload the annotations saved in the image’s annotation file.
4. Create pop up warnings before Remove All function is applied to have the user confirm the action first.
5. Create pop up warnings before Ctrl-Q function is applied to have the user confirm the action first. Ctrl-Q closes the application without saving the current image’s annotations.
6. Use color to highlight labeled segments.
7. Use different colors to highlight annotated segments with each object highlighted using a different color.
8. Preserve the superpixel boundaries when it is highlighted. When two adjacent superpixels are currently highlighted the boundaries between them disappear.